Notice of Allowability	Application No.	Applicant(s)	
	09/919,753	APUZZO ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *4/14/2005*.

2. ☒ The allowed claim(s) is/are *1-58*.

3. ☒ The drawings filed on *01 August 2001* are accepted by the Examiner.

4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
      1. ☐ Certified copies of the priority documents have been received.
      2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
      3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

6. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
      1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

7. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☐ Information Disclosure Statements (PTO-1449 or PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application (PTO-152)
6. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____ .
7. ☒ Examiner's Amendment/Comment
8. ☒ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 4/14/05.

As indicated in Applicant's response, claims 1, 15, 20, 34, 40, 41 and 51 have been amended, and claims 55-58 added. Claims 1-58 are pending in the office action.

EXAMINER'S AMENDMENT

2. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Kevin Radigan (Reg # 31789) on 5/27/2005.

The application has been amended as in the following.

In the CLAIMS (submitted with response filed 4/14/05), amend as follows:

1. A method of testing a software component comprising multiple layers of software, the method comprising:

creating an abstraction matrix in mathematical abstract form by automatically partitioning the software component into multiple layers, the abstraction matrix comprising state and event information taking into account relationships that exist between the multiple software layers;

parsing the abstraction matrix to automatically generate and factor out [[doable]] test cases and mapped expected results therefor;

separating the test cases based on the software layers of the software component, and associating data structures with the separated test cases of the software layers which allow~~, the data structures allowing~~ the test cases of the various software layers to be uncorrelated, wherein for at least one layer, the data structures simulate inputs to the layer from at least one other layer of the multiple software layers;

employing the software component in executable form to generate for each software layer of the software component test case execution threads from the test cases and mapped expected results for that software layer; and

executing in parallel at least some of the test case execution threads for at least one software layer of the software component, thereby testing the software component.

9. The system of claim 1, further comprising means for providing attributes for use in generating the test cases and mapped expected results therefor[e].

15. A computer-implemented method of ~~generating test cases for use in creating~~ testing a software component, the software component comprising multiple layers of software, said method comprising:

ascertaining a functional specification of the software component;

creating an abstraction matrix in mathematical abstract form that describes the software component by automatically partitioning the software component into multiple software layers using the functional specification, the abstraction

matrix comprising state and event information taking into account relationships that exist between the multiple software layers;

parsing the abstraction matrix to automatically generate and factor out [[doable]] test cases and mapped expected results therefor; [[and]]

separating the test cases based on the software layers of the software component, and associating data structures with the separated test cases of the software layers~~, the data structures allowing~~ which allow the test cases of the various software layers to be uncorrelated, wherein for at least one layer, the data structures simulate inputs to the layer from at least one other layer of the multiple software layers;

employing the software component in executable form to generate for each software layer of the software component test case execution threads from the test cases and mapped expected results for that software layer; and

executing in parallel at least some of the test case execution threads for at least one software layer of the software component, thereby testing the software component.

20. A system for testing a software component comprising multiple layers of software, the system comprising:

an abstraction matrix in mathematical abstract form that describes the software component, the abstraction matrix partitioning the software component

into multiple software layers and comprising state and event information taking into account relationships that exist between the multiple software layers;

means for parsing the abstraction matrix to automatically generate and factor out [[doable]] test cases and mapped expected results therefor;

means for separating the test cases based on the software layers of the software component, and for associating data structures with the separated test cases of the software layers~~, the data structures allowing~~ which allow the test cases of the various software layers to be uncorrelated, wherein for at least one layer, the data structures simulate inputs to the layer from at least one other layer of the multiple software layers;

means for employing the software component in executable form to generate for each software layer of the software component test case execution threads from the test cases and mapped expected results for that software layer; and

means for executing in parallel at least some of the test case execution threads for at least one software layer of the software component, thereby testing the software component.

28. The system of claim 20, further comprising means for providing attributes for use in generating the test cases and mapped expected results therefor[e].

34. A computer-implemented system for testing a software component comprising multiple layers of software, the system comprising:

an abstraction matrix in mathematical abstract form that describes the software component, the abstraction matrix partitioning the software component into multiple software layers and comprising state and event information taking into account relationships that exist between the multiple software layers;

a computer-implemented abstraction engine for parsing the abstraction matrix and automatically generating and factoring out [[doable]] test cases and mapped expected results therefor;

wherein the abstraction engine separates the test cases based on the software layers of the software component, and associates data structures with the separate test cases of the software layers~~, the data structures allowing~~ which allow the test cases of the various software layers to be uncorrelated, wherein for at least one layer, the data structures simulate inputs to the layer from at least one other layer of the multiple software layers; [[and]]

a computer-implemented functional verification test engine adapted to take the software component in executable form and generate for each software layer of the software component test case execution threads from the test cases and mapped expected results for that software layer, wherein the functional test verification engine outputs in parallel at least some test case execution threads for at least one software layer of the software component, thereby allowing testing of the software component; and

means for executing in parallel at least some of the test case execution threads for at least one software layer of the software component, thereby testing the software component.

35. A system for ~~generating test cases for use in~~ testing a software component, the software component comprising multiple layers of software, said system comprising:

a functional specification of the software component;

an abstraction matrix in mathematical abstract form that describes the software component, the abstraction matrix partitioning the software component into multiple software layers and comprising state and event information taking into account relationships that exist between the multiple software layers, and having been created from the functional specification;

means for parsing the abstraction matrix to automatically generate and factor out [[doable]] test cases and mapped expected results therefor; [[and]]

means for separating the test cases based on the software layers of the software component, and for associating data structures with the separated test cases of the software layers~~, the data structures allowing~~ which allow the test cases of the various software layers to be uncorrelated, wherein for at least one layer, the data structures simulate inputs to the layer from at least one other layer of the multiple software layers;

means for employing the software component in executable form to generate for each software layer of the software component test case execution threads from the test cases and mapped expected results for that software layer; and

means for executing in parallel at least some of the test case execution threads for at least one software layer of the software component, thereby testing the software component.

40. A computer-implemented system for ~~generating test cases for use in~~ testing a software component comprising multiple layers of software, the system comprising:

a storage medium for storing an abstraction matrix in mathematical abstract form that describes the software component by partitioning the software component into multiple software layers, the abstraction matrix, which comprises state and event information taking into account relationships that exist between the multiple software layers, having been created from an automatic partitioning of a functional specification of the software component;

a computer-implemented abstraction engine for automatically retrieving and parsing the abstraction matrix to automatically generate and factor out [[doable]] test cases and mapped expected results therefor; [[and]]

wherein the abstraction engine separates the test cases based on the software layers of the software component and associates data structures with the separated test cases of the software layers~~, the data structures allowing~~ which allow the test cases of the various software layers to be uncorrelated, wherein for at least one layer, the data structures simulate inputs to the layer from at least one other layer of the multiple software layers;

employing the software component in executable form to generate for each software layer of the software component test case execution threads from the test cases and mapped expected results for that software layer; and

executing in parallel at least some of the test case execution threads for at least one software layer of the software component, thereby testing the software component.

41. At least one program storage device readable by machine, tangibly embodying at least one program of instructions executable by the machine to form a method for testing a software component comprising multiple layers of software, the method comprising:

storing an abstraction matrix in mathematical abstract form that describes the software component by partitioning the software component into multiple software layers, the abstraction matrix comprising state and event information taking into account relationships that exist between the multiple software layers;

parsing the abstraction matrix to automatically generate and factor out [[doable]] test cases and mapped expected results therefor;

separating the test cases based on the software layers of the software component, and associating data structures with the separated test cases of the software layers~~, the data structures allowing~~ which allow the test cases of the various software layers to be uncorrelated, wherein for at least one layer, the data structures simulate inputs to the layer from at least one other layer of the multiple software layers;

employing the software component in executable form to generate for each software layer of the software component test case execution threads from the test cases and mapped expected results for that software layer; and

executing in parallel at least some of the test case execution threads for at least one software layer of the software component, thereby testing the software component.

51. At least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine to perform a method for ~~generating test cases for use in~~ testing a software component comprising multiple layers of software, the method comprising:

storing an abstraction matrix of the software component in mathematical abstract form that describes the software component by partitioning the software component into multiple software layers, the abstraction matrix comprising state and event information taking into account relationships that exist between the multiple software layers;

parsing the abstraction matrix to automatically generate and factor out [[doable]] test cases and mapped expected results therefor; [[and]]

separating the test cases based on the software layers of the software component, and associating data structures with the separated test cases of the software layers~~, the data structures allowing~~ <u>which allow</u> the test cases of the

various software layers to be uncorrelated, wherein for at least one layer, the data structures simulate inputs to the layer from at least one other layer of the multiple software layers;

employing the software component in executable form to generate for each software layer of the software component test case execution threads from the test cases and mapped expected results for that software layer; and

executing in parallel at least some of the test case execution threads for at least one software layer of the software component, thereby testing the software component.

55. The method of claim 1, wherein the data structures for at least some software layers [[mimic]] simulate information passed between layers during normal operation of the software component.

EXAMINER'S STATEMENT OF REASONS FOR ALLOWANCE

3. Claims 1-58 are allowed.

The following is an examiner's statement of reasons for allowance:

The prior art taken separately or jointly does not suggest or teach the following features.

A system or method for testing software component including generating an abstraction matrix comprising state and event information taking into account relationships that exist between the multiple software layers; and comprising

(i) parsing the matrix to automatically generate and factor out test cases and mapped expected results therefor;

(ii) separating the test cases based on the software layers of the software component, and associating data structures with the separated test cases of the software layers which allow the test cases of the various software layers to be uncorrelated, wherein for at least one layer, the data structures simulate inputs to the layer from at least one other layer of the multiple software layers;

(iii) employing the software component in executable form to generate for each software layer of the software component test case execution threads from that layer test cases and executing in parallel at least some of the test case execution threads; all (i), (ii), and (iii) as recited in claims 1, 15, 20, 34, 35, 40, 41, and 51.

Passova, USPN: 6,671,874, teaches developing a software system with a matrix to map system requirements conditions and events as a set of Petri net type of transitions and positions using additional variables or tags to support the conditions under which the transitions occur but Passova does not teach partitioning into software layers and parsing of the matrix to automate test cases and mapped results from the parsing as in (i); nor does Passova generate structures within the layer test cases that simulate in-between layer inputs as in (ii) and generating of layer test case threads for layer execution of thread in parallel as in (iii).

Okayasu, USPN: 5,659,554, teaches generating test cases via generating transition diagrams with flag setting as Passova for characterizing conditions for parallel

transitions; but fails to teach or suggest the sequence of (i), (ii) and (iii) as set forth above.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Conclusion

4. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence – please consult Examiner before using) or 703-872-9306 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT

May 27, 2005

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100